Linking and Combining Distributed Operations Facilities Using NASA's "GMSEC" Systems Architectures

Dan Smith[1], Thomas Grubb[2] and Jaime Esper[3]
National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, Maryland 20771, USA

NASA's Goddard Mission Services Evolution Center (GMSEC) ground system architecture has been in development since late 2001, has successfully supported eight orbiting satellites and is being applied to many of NASA's future missions. GMSEC can be considered an event-driven service-oriented architecture built around a publish/subscribe message bus middleware. This paper briefly discusses the GMSEC technical approaches which have led to significant cost savings and risk reduction for NASA missions operated at the Goddard Space Flight Center (GSFC). The paper then focuses on the development and operational impacts of extending the architecture across multiple mission operations facilities.

Nomenclature

API	=	Application Programming Interface
COTS	=	Commercial Off-the-Shelf
DOD	=	Department of Defense
GMSEC	=	Goddard Mission services Evolution Center
GSFC	=	Goddard Space Flight Center
MOC	=	Mission Operations Center
NASA	=	National Aeronautics and Space Administration
ORS	=	Operational Responsive Space
SOA	=	Service Oriented Architecture
AFRL	=	Air Force Research Laboratory (AFRL) and the Naval Research Laboratory (NRL)
SMEX	=	NASA's series of Small Explorer Satellites
ST5	=	NASA's Space-Technology 5 constellation of three satellites
TRMM	=	NASA's Tropical Rainfall Measuring Mission
XML	=	Extensible Markup Language

I. Introduction

NASA's Goddard Space Flight Center (GSFC) in Greenbelt, Maryland, USA operates most of NASA's sub-orbital and low-earth orbit unmanned scientific spacecraft. GSFC manages a wide variety of missions, from balloon and sounding rocket experiments to flagship missions like Hubble Space Telescope. Mission durations for orbiting spacecraft have ranged from several months to over 20 years.

About 30 satellites are managed by GSFC at any time, with about half of these

[1] GMSEC Project Manager, Software Engineering Division, Mail Stop 580
[2] Senior Engineer, Software Engineering Division, Mail Stop 583
[3] Senior Systems Engineer, Systems Engineering and Advanced Concepts Division, Mail Stop 592

operated from mission operations centers (MOCs) on the Greenbelt campus. Others are managed by universities across the United States and suborbital missions are managed from GSFC facilities at Wallops, Virginia, USA.

Historically, the missions each have had their own mission control center. Each mission was given full latitude to develop its mission control system and its operations plans as it wanted, with software commonality between missions being limited. At least five different telemetry and command systems were in use. Most software was developed by GSFC employees and their contractors. System designers often used the tools they were most familiar with from previous missions. Although some very advanced new tools were developed for specific missions, they often were not compatible with other missions systems.

Several problems emerged from the approach of having each mission responsible for development of its own mission operations center. Innovation was slowed as each budget-constrained mission worked to meet only its own requirements and to minimize additional system enhancements. Studies, designs and implementations efforts were often repeated for each new mission. COTS product lines were typically not even considered because they had never been integrated with other GSFC products.

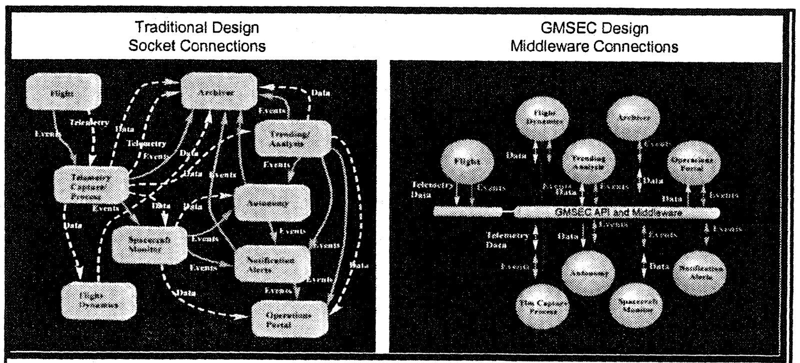
The problems with the traditional approach were made worse with the increased cost pressures of the late 1990's and the outlook for a healthy set of planned missions. The decision was made in 2001 to begin development on reference architecture for future missions. The new paradigm would represent a significant change from the previous approach of integrating selected components to create mission-unique systems. The new architecture, named the Goddard Mission Services Evolution Center (GMSEC) ground system, has been in development since late 2001 and has now successfully supported eight orbiting satellites. It is being applied to many of NASA's future missions and is being used outside of the control center environment.

The use of GMSEC has led to many observed benefits ranging from shorter development and integration times to increased automation and efficiency of the operations efforts. It has also led to a natural progression of user requests for additional capabilities, with many of the requests involving new interfaces to other GMSEC-based and non-GMSEC-based operations centers. Work is now underway to develop more standardized approaches for interoperability between these systems.

II. The Goddard Mission Services Evolution Center (GMSEC) Architecture

A. High-Level Architecture Concepts

GMSEC can be considered an event-driven service-oriented architecture built around a publish/subscribe message bus middleware. The GMSEC Applications Programming Interface (API) allows any of several different middleware packages to be utilized simultaneously. Standard message formats for common interfaces allow for simplified integration of many of today's most popular commercial products. Situational awareness and automation tools take advantage of the ability to communicate with any of the functional subsystems on the message bus.



The GMSEC API provides the generalized interface between applications programs and the system middleware. Multiple languages, middleware products, platforms and operating systems are supported.

The GMSEC system was developed with several key architecture principals:

1. **Standardize Interfaces** – not Components. GMSEC does not perform trade studies and make component recommendations or selections. By standardizing interfaces, GMSEC encourages the

access to a broad selection of tools and the development of new and innovative products.

2. **Middleware Infrastructure.** At the heart of the architecture is a message oriented middleware (MOM). It can be a commercial package, open source or GSFC-developed.

3. **User Choice.** GMSEC is not trying to select the best component in each functional area. The GMSEC team is not trying to compare COTS products against each other or against a heritage system. Instead, the architecture allows the user to select the most appropriate products based on functional need or personal preference and easily integrate them into a ground system. Keeping this responsibility with the mission teams has greatly increased the acceptability of the GMSEC approach.

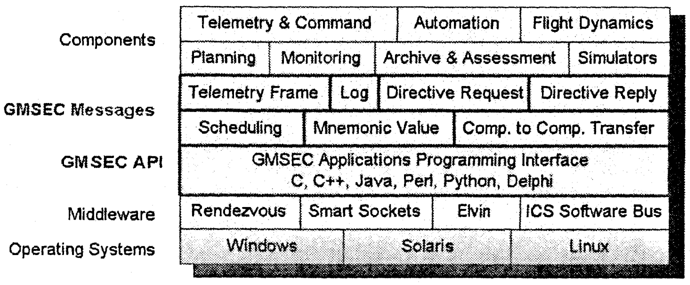
4. **General-Purpose Approach with Flight-Ground Capabilities.** The architecture itself should be designed to be adaptable to any number of missions. The key concepts should be applicable to either flight software or ground control systems and should, ideally, be extensible to other domains.

The GMSEC architecture is not a full formal Service Oriented Architecture (SOA). It does not include a full dynamic registry of available services. GMSEC does, however, have attributes to meet some of the goals of a SOA. The message-oriented middleware keeps track of the locations of the software components so hard-code node routing is not needed and logical/physical location transitions can occur instantly in the case of failovers. The use of an API separates the applications code from the underlying details. The use of message standards, adapters and the available API allows for the integration of existing or newly developed components. The message adapters allow COTS components to be used without changes.

B. The Message Bus and Standard Message Formats

The GMSEC Architecture uses a message bus (sometimes called an information bus or software bus) for inter-process and inter-node communication. Instead of traditional socket connections among components, each component only interfaces with the message bus. The middleware keeps track of where processes are located and which process requires the data when it is published to the bus.

The message bus provides publish/subscribe message passing mechanisms. Applications "publish" messages to the bus. Each message contains a subject name and the normal message contents. The subject name, for GMSEC applications, indicates the mission, originating node, type of message, etc. Applications that need the data "subscribe" to the pertinent subject name(s) and the middleware delivers the messages which match the subscribe request. Although the publish/subscribe mechanisms are common to many different middleware products, each product uses its own proprietary message structure for passing the data on the bus. The commercial middleware products are therefore not compatible with each other and applications are normally written to match the specific middleware package selected for the system development effort. The GMSEC API normalizes the basic capabilities of multiple middleware products so they each appear the same to the applications software. In this way, a change to the middleware product does not require a change to the applications software and vendor lock-in is avoided. This middleware flexibility allows for product swapping if necessary, but also allows for low-cost middleware to be used for development, high-reliability and high-performance middleware to be used for operations and small-footprint middleware to be used for flight – all with the same functional behavior.



The GMSEC API provides the generalized interface between applications programs and the system middleware. Multiple languages, middleware products, platforms and operating systems are supported.

The GMSEC system, with its middleware, can transfer 10's of megabits of data per second – a rate 100's of time beyond what missions today typically require for command and control and health and safety data. Typical overhead of the GMSEC middleware is less than 2% of the CPU.

The GMSEC API provides isolation between the applications programs and the underlying messaging software. As discussed above, any of several different middleware packages can be used without modifying the applications. In addition, the API supports multiple languages, operating systems and platforms. It normalizes the behavior of the middleware while allowing access to special functions or capabilities of individual middleware products.

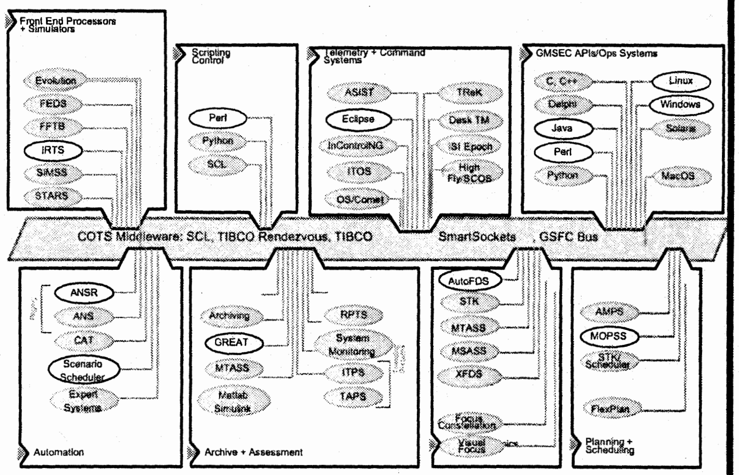
GMSEC standard messages meet the needs of the key interfaces for mission control applications. Additional messages may be created as needed. A subject name is specified with each message published via the API. The subject is used for routing to the subscribing applications and contains information including the mission identifiers, nodes, message type (e.g., telemetry), etc. The messages themselves include a common header used by all messages followed by a message-specific body. The message header contains some of the same subject information, but also includes time stamps and more details on the message type. The body of the message may contain a telemetry frame or packet, a text message, user directive, archive request, etc.

Documentation from many commercial products and from over a dozen GSFC in-house systems were examined to create standard message formats which would stand the test of time and allow for easy adaptation from messages which may be output by existing systems.

C. Compliant Components

GMSEC uses the common interface approach to allow many different products of the same functional domain area to be integrated. By having choices in each functional area, missions can avoid vendor lock-in and can select each component based on its own merits (technical, cost, etc.). Components can be as major as telemetry and command systems or planning systems or as small as performance monitoring tools.

An "adapter" approach is used for the existing in-house or commercial components. The adapter is a piece of software which works like an API-to-API interface and converts from the existing package's interfaces to the GMSEC interfaces. Because the GMSEC interfaces were developed with knowledge of many COTS interface definitions, this adaptation has been proven to go very quickly (from a day to about 2 weeks). COTS packages that do not have clean, exposed APIs are more difficult to adapt and their underlying design may not be amenable to GMSEC adaptation. Obviously, new software can be written to directly use the GMSEC API calls.



GMSEC Catalog. Choices are available for many subsystems. Nearl all COTS command and control systems are now GMSEC compatible [acronyms not important]

Each major component is required to meet certain standards to be considered "GMSEC compliant":

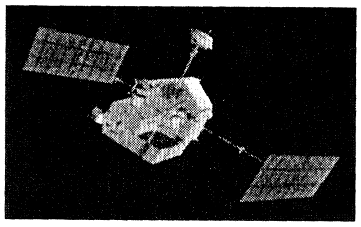
1. It must meet its functional requirements; although verification responsibility is with the missions
2. It should publish a heartbeat message on a periodic basis
3. It should publish status/log messages to indicate an action has taken place or an event has occurred
4. It should allow user directives for the component's control to be received over the message bus

These simple rules can yield very powerful results. The heartbeats allow for system monitoring, configuration displays and failovers. The log messages across multiple tools provide a new level of situational awareness. Allowing directives to be sent to any component allows for scripting and, combined with situational awareness, provides new levels of reactive system automation.

A typical GMSEC control center includes a simulator, front-end processor, telemetry and command system, trending system, automation tools, scheduling tools and a set of general support and performance tools. The GMSEC catalog of available components lists several choices in each of these categories. The end result is a set of GMSEC-based control centers, each with the same basic architecture and some common tools, but also with differences in some of the key components selected.

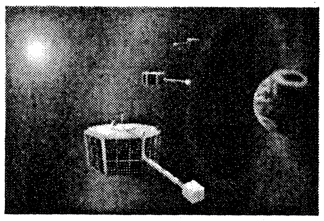
D. Current Use and Observed Benefits

GMSEC-based mission operations systems at GSFC have been operational since 2005 and most of GSFC's major satellite systems now in development will be using the GMSEC core capabilities.



TRMM was the first mission to apply the GMSEC technologies. By basing their reengineering effort around the GMSEC Architecture and concepts, the TRMM mission was able to add significant levels of automation and move to fewer operational shifts per day without any science data loss. The annual operating budget was reduced by about 50% and the reengineering efforts were fully paid for out of incremental budget savings. The reengineered system has paid for itself within 2 years.

The Small Explorer (SMEX) missions, including the TRACE, SWAS and Wire satellites implemented the GMSEC architecture to enable new capabilities associated with satellite fleet operations and constellation operations. The system has successfully demonstrated continuous "lights-out" operations. No direct operator involvement in the control room is required – the system pages users if a problem is detected. The SMEX reengineering effort is considered a pathfinder for future GSFC low-cost fleet operations and the operations involving of satellite constellations.



The ST5 mission was a 90-day operation consisting of a constellation of three small satellites launched in March 2006. Its control center was used to support satellite integration, pre-launch checkout, training, and on-orbit operations. Power and solid state recorder subsystems were modeled using Matlab/Simulink within a GMSEC-compliant component. Real-time telemetry was used to update the model so that predictions could be made based on scheduled future activities. If problems were projected, the modeling system notified the scheduling system to make adjustments. ST5 reported that GMSEC saved them money while allowing them to demonstrate more advanced capabilities. ST-5 also demonstrated automated "lights out" automation with GMSEC for a full 2-week operations period with the system paging the operations team as needed.

The GMSEC architecture is also being applied to reengineering efforts and new development efforts for 5 GSFC control centers currently under development. It has also been used in labs across other NASA Centers and is operational at the Marshal Space Flight Center. Some of the key benefits realized by the developers and their missions include the following:

1. There has been a significant reduction in integration time
2. Components can be added/upgraded without impacting the existing system
3. The message bus approach is ideal for using multiple small distributed development teams and vendors
4. New concepts are emerging for small independent components that integrate with the bus and provide immediate benefits
5. Missions are more willing to adopt the approach if "old favorite" components can still be used
6. Some vendors see message compliance as a way to enter what had appeared to be a closed marketplace
7. The standard message approach provides collaboration possibilities with other organizations

8. Automation for cost and risk reduction is the #1 selling point

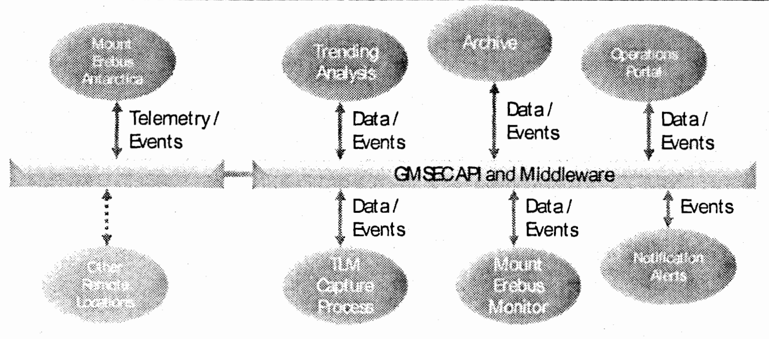
E. Future Application Example: Operational Responsive Space (ORS)

In addition to being planned for specific NASA missions, the GMSEC architecture is also being evaluated for use on entire ranges of missions within NASA and for other large space organizations. Operational Responsive Space (ORS) is one such example.

"Responsive Space" refers in general to the ability to bring on-line systems in a rapid or accelerated fashion. "Operational Responsive Space" refers to such systems used to satisfy the *current* needs of either NASA applications (e.g., astronaut health and safety requiring immediate attention), or Department of Defense (DOD) applications (e.g., need to deploy strategic assets in short order). Since 2003 NASA GSFC has been working with the DOD on space flight systems that feature modular plug-and-play interface standards as a means of enabling rapid integration and test, system cross-compatibility, and reduced costs (Refs: 1 and 2). Concurrent developments by DOD laboratories such as the Air Force Research Laboratory (AFRL) and the Naval Research Laboratory (NRL) have seen considerable progress in recent years. Although the evolution of modular, reconfigurable systems can be traced back at least 38 years to the Multi-Mission Modular Spacecraft and other GSFC programs and missions, the GMSEC architecture provides a recent and successful application of generalized principles of multi-use to the data system domain, and specifically to mission operations. It also represents an architecture and framework of high relevance to the current DOD ORS formulation.

Up to this date, ORS has concentrated on developing flight systems (modeling and simulation, spacecraft, launch vehicles) that feature rapid delivery from concept to operations. Less has been done in expanding on the fundamental need to "operate" such systems efficiently, incorporating the latest application software components quickly, and serving the operational need of distributed systems either on the field or in space. GMSEC provides such capability *today*. One of the main premises of ORS is to simplify the rapid infusion of technology over time. By concentrating on the interface and normalizing the capabilities of multiple middleware products, GMSEC allows the incorporation of software components and applications in an efficient and rapid fashion, enabling the acquisition of "new" data as quickly as the application is ready or as quickly as it can be acquired from the commercial market as a COTS product. There is no need for extensive testing of a proprietary message structure. Simply, "plug-and-play" or "configure and operate". Many different products can operate concurrently in the GMSEC Lab environment; creating a new mix of products can be as simple as turning off the ones not needed. In addition to preventing data system obsolescence, GMSEC also enables the rapid delivery of information across its framework.

The following application example highlights the operational benefits of GMSEC. Although this scenario is geared toward a civilian time-critical application, it is also equally applicable to the military and ORS. In this scenario a team of scientists in Antarctica require the use of the vast facilities of the home organization for analyzing seismic data from Mount Erebus. The equipment on the field has failed to operate as planned and the only surviving hardware at their disposal is a seismometer, a laptop, a transponder, and a satellite link. This scenario is similar to having a "remote service" request to the MOC. In the GMSEC framework, connectivity and use of the home organization's analytical tools is as simple as connecting the seismometer to the laptop, and sending the information for further real-time analysis. Warnings of an impending eruption can be issued to the team at the base of the caldera



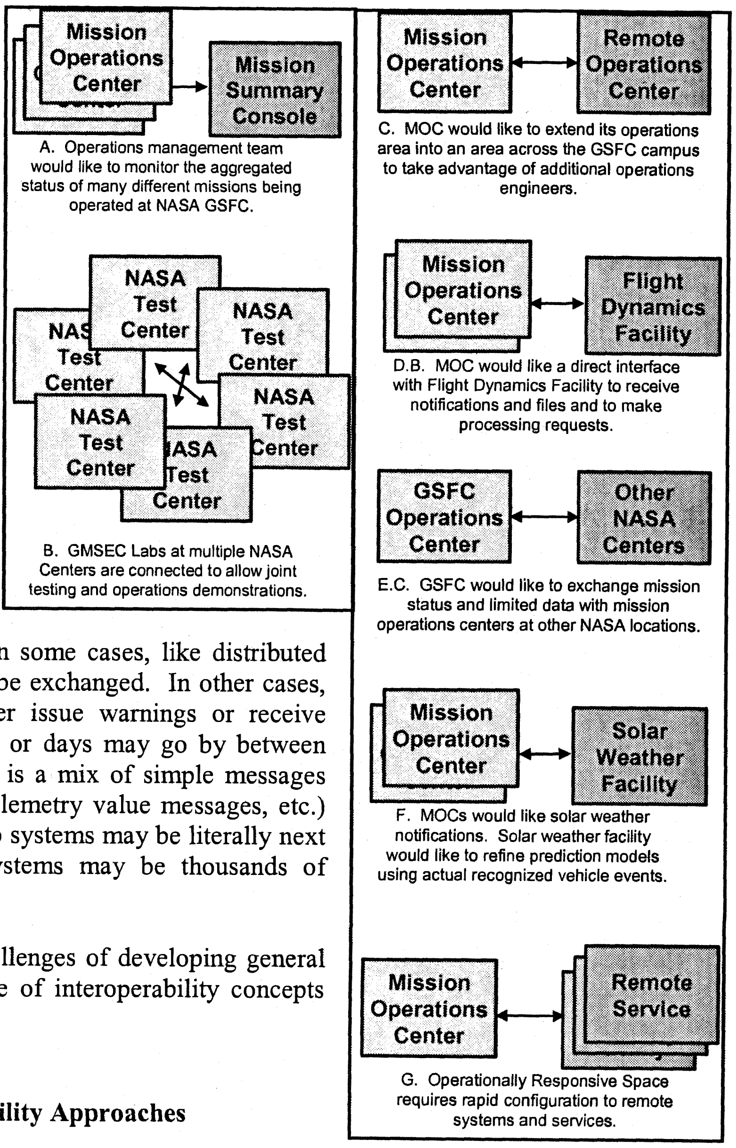
GMSEC enables the rapid, flexible integration of remote locations, leverages new or existing applications, and allows for high-speed data transfers through its software bus, all features desirable to ORS.

with advance time for safe evacuation. Additional application components can be brought on-line as quickly as required to interpret the results from the field. Better yet, other remote locations can be added to the message bus middleware as needed to enhance the data acquisition. This scenario is illustrated in the figure below.

III. Expanding Beyond the Initial Boundaries

A. Requirements for Interoperability and New Interfaces

As GMSEC has matured, the types of new capabilities for future systems have changed. The basic component-based concepts and use of COTS have been proven operationally. The automation tools have reduced the operations costs. Each mission operations center, however, is generally still run as its own system, with external interfaces being kept to a minimum and often very labor-intensive to work with. Many of the new requests now being received by the GMSEC team seek to create new or better connections between mission operations centers and either other centers or facilities. Although the requests appear as functionally independent ideas, many actually have much in common with each other.



A. Operations management team would like to monitor the aggregated status of many different missions being operated at NASA GSFC.

B. GMSEC Labs at multiple NASA Centers are connected to allow joint testing and operations demonstrations.

C. MOC would like to extend its operations area into an area across the GSFC campus to take advantage of additional operations engineers.

D.B. MOC would like a direct interface with Flight Dynamics Facility to receive notifications and files and to make processing requests.

E.C. GSFC would like to exchange mission status and limited data with mission operations centers at other NASA locations.

F. MOCs would like solar weather notifications. Solar weather facility would like to refine prediction models using actual recognized vehicle events.

G. Operationally Responsive Space requires rapid configuration to remote systems and services.

The diagrams to the right show some of the interconnection requests received by the GMSEC team in the past year, some of which have been developed as single-use solutions. In some cases, like distributed operations concepts, high volumes of data must be exchanged. In other cases, such as having the GSFC solar weather center issue warnings or receive problem indications from control centers, hours or days may go by between message exchanges. For flight dynamics, there is a mix of simple messages (product request and product ready messages, telemetry value messages, etc.) and product file transfers. In some cases, the two systems may be literally next door to each other and in other cases the systems may be thousands of kilometers away.

The sections below describe some of the challenges of developing general purpose interface capabilities to cover the range of interoperability concepts now being considered.

B. Industry Trends Affect Interoperability Approaches

Interoperability challenges are easy to resolve. All that is needed is agreement for every mission control center and every supporting system and facility to use the same architecture, software, messages, protocol, security and operations concepts.

Clearly, the world is not headed towards a single ground system implementation approach. There will always be a mix of old and new systems. There are several very positive trends taking place in the ground system industry which affect the handling of interfaces:

1. COTS vendors now have well defined interfaces for accessing different message types.
2. Message-oriented middleware such as GMSEC have introduced the concepts of robust standard message formats adaptable to many different products and missions.
3. The Object Management Group has established standards for publish/subscribe middleware and for CORBA
4. The CCSDS standards committee is working on a ground system Service Oriented Architecture standard for use by space agencies around the world
5. NASA, as part of its Exploration Initiative, is investigating the concepts of common approaches to enable interoperability across many of the future NASA systems.
6. Many major aerospace companies are developing advanced Enterprise systems and Service Oriented Architectures for space mission control

Individually, these ideas represent significant advances focused on simplified integration, common tools and interoperability. Collectively, however, these ideas are creating a lack of commonality in the name of commonality! Additional architecture and design work is needed to incorporate concepts to enable interoperability between similar and dissimilar systems. More often than not, remote systems are not constructed the same way as the local system.

C. Connectivity and Interoperability Factors

The following factors should be considered when developing solutions for connecting independent systems for the purpose of exchanging known data types:

1. Can not simply design the new best architecture and apply it to all systems and users.

2. Can not expect to join multiple Service Oriented Architecture systems using a shared master registry.

3. Can not declare a new universal standard – other can not be expected to follow.

4. Can not disregard the different security requirements associated with each system or even that security requirements will remain stable.

5. Can not assume that the internet is the answer – many networks for satellite control are not open to the public networks

6. Must allow for the configuration of routing and data filtering parameters to control the interface operation

7. Must assume a wide mix of remote system designs

8. Must assume that other systems are operational – special care is needed

9. Must allow for either end of an interface to disconnect in case of problems

10. Must assume there will be sensitivities to what data can be shared

11. Must be low cost, high dependability

IV. Solutions for Linking and Combining Operations Facilities

GMSEC enables easy *inter-application* communication to make powerful systems of interconnected applications. However, a system is rarely operating in a vacuum. Linking and combining operations facilities is a vital part of expanding enterprise architectures. This *inter-system* communication poses unique challenges. Inter-system communication must not only solve the technical challenge of communication (e.g., data translation, data exchange and protocols) but must deal with issues dealing with trust, security, and control.

Do two systems trust each other? Is the problem really just one of making the two systems talk and act like one big virtual system? Does every application have the same privileges with respect to other applications regardless of which system they are located? This is the simplest case. However, often each system is an island, and the system managers want to control the access and interchange of information. Perhaps, instead of a two-way communication, really all that is wanted is one-way stream to publish warning events to external systems. These cases require a more nuanced solution.

Within the GMSEC project, there have been two basic techniques for inter-system communication: adapters and bridges. These have been specific case-by-case solutions in the past, and the GMSEC team is now developing general purpose interface capabilities to meet the new functional requests.

A. Virtual Networks (Middleware Extension)

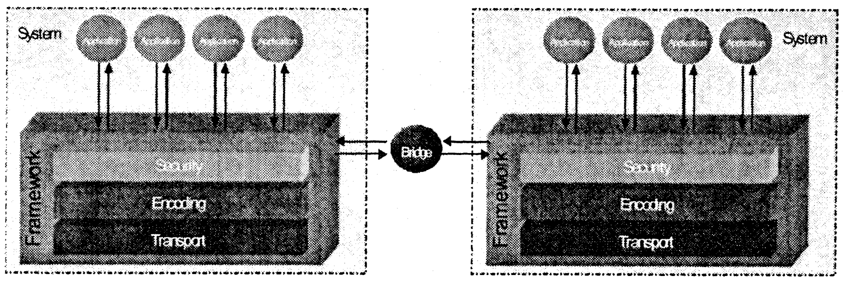
The simplest case for inter-system communication is creating a virtual network that comprises two (or more) systems. The goal is to create a one large virtual system, that is comprised of two or more physically separated systems. Every application acts like it is connected to one middleware system, not differentiating between the two systems.

With two GMSEC systems that use the same middleware, it may be possible to extend the system network virtually. Tibco SmartSockets allows connecting the servers of two systems so that they act as one system. Using Tibco SmartSockets, GMSEC has successfully created a virtual network between two operation centers to make one virtual system. Both physically separated systems would publish messages normally, subscribe normally, and receive messages normally. In this case, for the applications of the systems, it truly appears like one system.

GMSEC has also used this technology to create networks between multiple NASA centers. GMSEC systems were installed at five different NASA Centers as part of a coordinated Exploration Initiative test effort. Communications over both NASA and public networks were established and collaborative tests were conducted to show distributed test and operations concepts. The entire effort took place on one large virtual system.

B. Inter-GMSEC Bridging

GMSEC has also bridged between two or more GMSEC systems that use different middleware products. GMSEC created a bridge that can be configured to connect to any two GMSEC systems, regardless of their middlewares. The bridge connects to each GMSEC system, subscribes to both systems, and passes messages between the systems (See Figure).



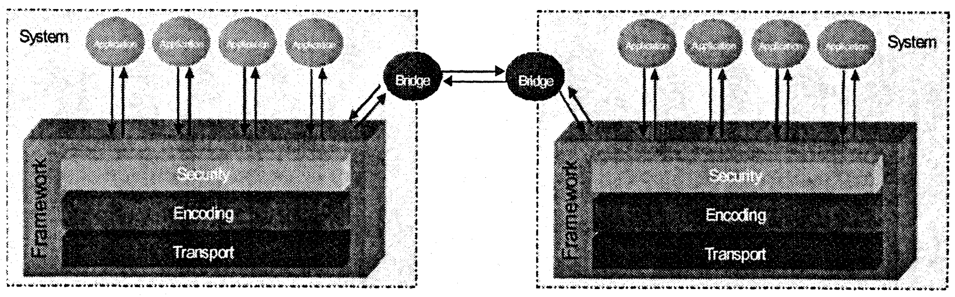
When the GMSEC bridge receives a message from one system, it unwraps the message, creates a new message on the other system, and then wraps and publishes the message to the other system.

Using configurations of subscriptions and filters, the GMSEC bridge is able to control what messages are passed through to each system. The bridge can create two-way streams or a one-way stream. The GMSEC Bridge is a generic solution for bridging two or more GMSEC systems.

Advanced Inter-GMSEC Bridging

There are a couple of issues with Inter-GMSEC bridging. The first issue is with performance. There is a performance penalty for unwrapping and wrapping the many individual fields that can make up a message. In a high-volume situation, this penalty can become prohibitive. However, there are ways to address this issue by treating the entire message contents as a single aggregated data field to avoid the field-by-field extraction and repackaging.

The bigger issue is with security and trust. If you use only one bridge between two systems, the bridge must contain all the security for decrypting and encrypting messages for *both* systems. As can be imagined, this super user application is a huge security vulnerability. It also requires a lot of trust between the two systems, as the keys are known between the systems.



A better bridging scenario is to provide two bridges, one bridge per system (See Figure). Each system is responsible for running one bridge within their system. The two bridges then connect and pass messages between them. This scenario has a number of advantages. The system now gets to control not only the security information but also the traffic into and out of the system. The local bridge can filter out many messages before passing them over a possibly slow connection to the other system. There is generally no need to send the large volume of local publish/subscribe traffic between systems. The two bridges need share only the keys necessary to pass information securely between each other.

Of course, there are issues with this scenario as well. First, the local bridges are still super user applications, with access to all the keys of the local system (something possibly not desired by the system builder). Also, two applications cannot talk privately, there is always an interpreter (the bridges) between them. In some cases, it may not be possible for an application to sign its communications because the signing keys are not shared. Finally, there is the performance penalty, the unwrapping and wrapping messages has been doubled over the single bridge case.

C. GMSEC to Other Middleware Bridging

When you need to bridge between GSMEC and some other middleware and non-GMSEC systems , the bridging becomes more difficult. A unique solution may need to be created. GMSEC has experience with this type of bridging as well. For connecting a large data management system to a GMSEC mission control center, a one-way bridge was created that passed specific messages from the data management middleware onto a GMSEC bus. It didn't occur in this case, but *data translation* may also be an issue when exchanging messages between GMSEC and another system.

A GMSEC-to-Other designer must be careful to preserve the syntax and the semantic meaning between messaging systems. The GMSEC team is currently developing an adapter to connect the GMSEC mission operations center systems at GSFC to GSFC's solar weather science center and to another NASA operations center. In both cases, the communications can be limited to simple log/text messages. Even with this simple exchange, translations are needed between the standard GMSEC message format and the formats used by the other systems.

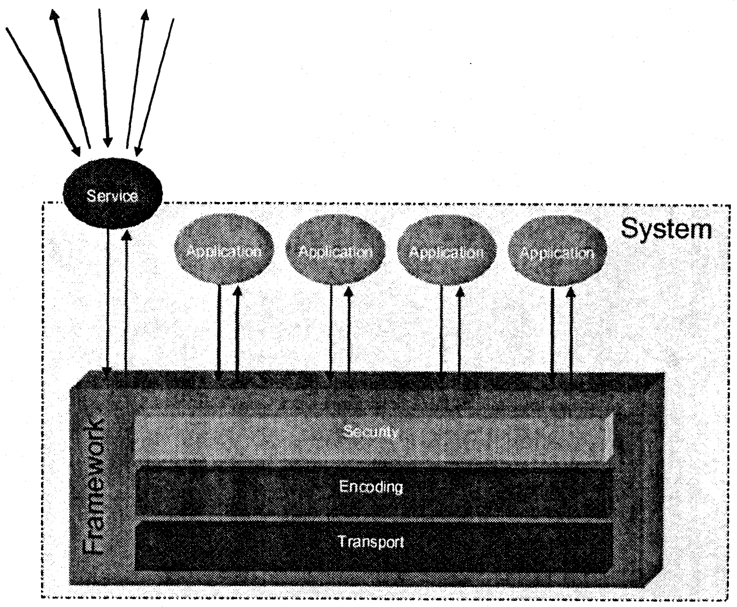
Another application was created that bridges a GMSEC bus with Web Services, Java RMI, and Corba ORBs. The adapter is responsible for passing messages back and forth between GMSEC and these other technologies. In this case, there are issues between the different messaging systems. For example, web services use essentially a polling mechanism. A web service consumer (which would call this adapter) needs to call the web service in order for any information to be exchanged. However, GMSEC is a publish and subscribe architecture, which uses a push mechanism (e.g., a publisher pushes the information onto the bus to be delivered to any application that has subscribed to it). The two mechanisms are very different and the applications that used the adapter needed to be aware of that.

In general, in order to bridge between GMSEC and any other middleware, a bridge application will need to be built for each non-GMSEC middleware. A designer must build an application that connects to GMSEC and their target system, subscribe to a subset of messages on both systems, and translate between the two.

GMSEC has considered ways to make a generic GMSEC-to-Other Bridge using XML. GMSEC is able to input and output any message as XML. If the other middleware also has this option, an XSLT could be created to transform messages between GMSEC format and the other middleware's format. Of course, the performance hit would be considerable, but it could be a possibility in low-volume connections.

D. System As A Service(s)

Another option for inter-system interoperability is to present the *system as a service*. With System-as-a-service, the system integrator builds an application that acts as one or more service interfaces to the system. This service interface application is responsible for receiving operation requests on the system, translating them into the system "language", communicating with the applications of the system to provide the operation results (usually using the framework of the native system), and then translating and sending the results back to the consumer.



For example, the system could create a telemetry service application. This service would have operations such as GetMnemonic, ConvertRawToEU, etc. When this service is called from an outside entity, the application converts the operation call into a message that it would publish on the GMSEC bus. The real telemetry provider would be subscribed to these messages and publish a response back. The telemetry service interface application would take the response from the bus and respond to the original operation call.

Note that there is not necessarily a one-to-one mapping between a service interface like telemetry and applications on the system bus. The service interface application can present a composite service interface which is a combination of functionality fulfilled by multiple applications in the system.

To the outside world, the service looks like one entity, the service (or group of services).

The System-as-a-service provides the system builders with the advantage that they completely control what functionality of the system is exposed to the outside world. In addition, any responsibility for data translation burdens can be passed onto the service consumer. Finally, the security of the system is encapsulated at the service level and the applications on the system bus don't have to worry about these concerns.

A disadvantage of the System-as-a-service is that it is relatively static. To expose new capabilities of the system requires changing the service interface and the service interface application. The system builder must also maintain and host the service interface application. Finally, the system acts only as a service *provider* (at the least, the external entities control initiation of the service). In order for the system to act as a service *consumer*, another solution must be found or the system must rely on the external entities to call the service and provide any data.

V. Conclusions

GMSEC is now a mature component- and message-based system which has been proven with support to eight satellites across three missions for the past three years. More systems will become operational in 2008. The standard message capabilities of GMSEC have simplified the integration of many commercial and GSFC-developed software components and have allowed other organizations to develop applications "per the message specification." Missions using the GMSEC architecture have benefited from reduced development and integration time, a greater selection of products to choose from, and reduced operations costs resulting from increased automation of routine activities.

The next logical extension of GMSEC is to allow interaction between GMSEC systems and other GMSEC- or non-GMSEC systems. Uses can be as simple as simple warning message exchanges or as complex as distributed operations across multiple sites using different system approaches. The advancement of service oriented architectures and emerging standards to support interoperability have actually complicated the efforts to interconnect systems because of the wide range of independent system implementations now underway. Although many external interaction requests can be generalized to the same basic problem, a single solution may not be practical. Depending on the data to be exchanged and the differences in the two systems, the solution may involve single or multiple bridging applications or the encapsulation of system capabilities into a service exposed for external access. One bridging approach could connect a GMSEC system with a web services application.

The GMSEC team has developed solutions for many different system interconnection requests and is continuing to work on reducing the number of solution types to create more flexible, general purpose system gateways that can be easily configured and deployed. The same challenges are being faced by all large system designers as the needs for interaction across broader and broader information communities continues to expand.

References

[1]Esper, J., "Modular, Adaptive, Reconfigurable Systems: Technology for Sustainable, Reliable, Effective, and Affordable Space Exploration, *Space Technology and Applications International Forum*, American Institute of Physics, 2005.

[2]Esper, J., Andary, J., Oberright, J., So, M., Wegner, P., Das, A., Hauser, J., "Modular, Reconfigurable, and Rapid Response Space Systems: the remote sensing advanced technology microsatellite," *2nd Responsive Space Conference*, El Segundo CA, 2004.

Madden, M., Cary, E. Jr., Esposito, T., Parker, J., Bradley, D., "Lessons Learned from Engineering a Multi-Mission Satellite Operations Center*", IEEE Aerospace Conference*, Big Sky Montana, 2006.

Smith, D., Bristow, J., Wilmot, J., "A Successful Component Architecture for Interoperable and Evolvable Ground Data Systems", *AIAA Space Ops 2006 Conference*, Rome, Italy.

Joshi, R. Real Time Innovations, Inc., "Data-Oriented Architecture: Loosely Coupling Systems into "System of Systems"", *RTC Magazine*, January 2008.